INTERNATIONAL STANDARD

ISO 20078-2

First edition
2019-02

Road vehicles — Extended vehicle (ExVe) web services —

Part 2: Access

Véhicule routiers — Web services du véhicule étendu (ExVe) —

Partie 2: Accès

Reference number
ISO 20078-2:2019(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 22, *Road vehicles,* Subcommittee SC 31, *Data communication*.

A list of all parts in the ISO 20078 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Road vehicles — Extended vehicle (ExVe) web services —

Part 2: Access

1 Scope

This document defines how to access Resources on a Web services interface of an Offering Party using the Hypertext Transfer Protocol Secure (HTTPS). For such an access, the Representational State Transfer (REST) architectural pattern is chosen as a common way to format Resource paths. Some specific extensions to this pattern are defined to allow for asynchronous Resource requests, such as, for example, forcing readouts of data from a connected vehicle.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 20078-1:2019, *Road vehicles — Extended vehicle (ExVe) web services — Part 1: Content*

ISO 20078-3, *Road vehicles — Extended vehicle (ExVe) web services — Part 3: Security*

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 20078-1 apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.2 Abbreviated terms

For the purposes of this document, the abbreviated terms given in ISO 20078-1 apply.

4 Representational State Transfer based Interface

4.1 General

The following defines the requirements on a Representational State Transfer (REST)[3] based web service interface using Hypertext Transfer Protocol Secure (HTTPS)[1][2][8] based on Transport Layer Security (TLS) to give the Accessing Party secure Access to Resources provided by the Offering Party.

REQ_04_01_01	The REST based web services interface implementation shall use the Hypertext Transfer Protocol Secure (HTTPS) as transport protocol with Transport Layer Security (TLS).

REQ_04_01_02	HTTP shall only be used with version 1.1 or higher compatible versions.

REQ_04_01_03	TLS shall only be used with version 1.2 or higher versions.

REQ_04_01_04	The REST web service shall be a strict client-server interaction, where the Accessing Party (client) sends a request and the Offering Party (server) sends a response.

NOTE Resources can be transferred both in the request and the response.

REQ_04_01_05	The REST implementation shall be stateless; i.e. the Offering Party server shall not maintain any Accessing Party client context or session information.

Due to REQ_04_01_05 each request-response pair are handled independently from one another. Each client request by the Accessing Party contains all information required by the server of the Offering Party to successfully respond to the request, including a representation of the client state when necessary.

4.2 Resources

REQ_04_02_01	Information on the server shall be exposed as Resources expressed as plural nouns.

NOTE 1 This holds true even when the Resource is only available one time on a connected vehicle (e.g. "odometers").

REQ_04_02_02	The exposed Resources shall be uniquely identified in the form of Uniform Resource Identifiers (URIs).

The Resources, the Resource Groups or the Containers, and how to apply those on a specific presentation or application layer of the Accessing Party are described in ISO 20078-1. How an Accessing Party authenticates and how it is authorized for Access to Resources is described in ISO 20078-3.

REQ_04_02_03	The Offering Party shall define the base URIs of the web services.

Table 1 — Examples of possible ExVe base URIs

Resource	Description
https://{example.com}/exve/	URI based on sub directory.
https://exve.{example.com}/	URI based on sub domain.

REQ_04_02_04	The Offering Party shall comply with the URI Resource paths defined by specific ExVe standard applications.

Table 2 — Examples of possible ExVe Resource URIs

Resource	Description
{base_URI}/{resourcePath}	Resources based on path.
{base URI}/vehicles	A list of the available vehicles for a specific user
{base URI}/vehicles/{vehicleId}/dtcReadouts/	Read all Diagnostic Trouble Codes for a specific vehicle
{base URI}/vehicles/{vehicleId}/ecus/{ecuId}/dtcReadouts/	Read all Diagnostic Trouble Codes for a specific ECU of a specific vehicle

There are two primary elements defining an URI; Entities and Resources (Table 2). Entities are the fundamental objects representing e.g. vehicles, ECUs, drivers and fleets. Resources are the actual data, aggregated information or functions associated with an entity and a specific use case.

REQ_04_02_05	Resources shall be named and described.

EXAMPLE Fuel level could be an example of a single data item Resource and vehicle position an aggregate consisting of several data items (e.g. latitude, longitude, sample time), lock and un-lock the vehicle a functionality.

REQ_04_02_06	The Offering Party should have the possibility to extend Resources, but shall not be able to reduce Resources.

Thus by REQ_04_02_06 it is not possible to remove data items from a Resource, other than through an update of the underlying use case specification. It is however possible to add data items to a Resource (i.e. versioning).

REQ_04_02_07	Aggregated Resources shall only include or cross-reference necessary data items for the complete and correct operation of the related use case.

NOTE 2 REQ_04_02_07 ensures an Accessing Party only receives data items necessary for fulfilling the intended need and nothing else when accessing the Resource (i.e. data economy).

See also ISO 20078-1:2019, 8.3, for further detailing.

REQ_04_02_08	When defining Resources care shall be taken not to redefine a Resource already defined by an existing use case.

An application may extend the recommended patterns below if none of them meets the needs of the use case.

REQ_04_02_09	Each use case shall define how the HTTP operations GET, POST, PUT and DELETE are supported for each defined Resource and what the response for each operation is.

REQ_04_02_10	All URI elements shall be written in lower camel case notation.

NOTE 3 The {baseURI} in following patterns refers to the Offering Party root URI (see Table 3).

Table 3 — Examples of base URI expresses REQ_04_02_10 and REQ_04_02_01

Normative generic format	{baseURI}/{entities}
Example	{baseURI}/vehicles Returns a list of all vehicles available to the Accessing party.

REQ_04_02_11	Relations of entities shall be expressed using sub-resources.

NOTE 4 See Table 4.

Table 4 — Examples of sub-URI's expresses REQ_04_02_11 and REQ_04_02_01

Normative generic format	{baseURI}/{entities}/{ID}/{entities2}/{ID2}
Example	{baseURI}/fleets/12/vehicles/456 Get information on vehicle with id 456 of fleet with id 12.
	{baseURI}/vehicles/456/ecus/789 Get information on ecu with id 789 of vehicle with id 456.

REQ_04_02_12	A Resource shall be placed after the entity to which it belongs in the URI.

NOTE 5 See Table 5.

Table 5 — Examples of descriptive URI's expresses REQ_04_02_12 and REQ_04_02_01

Normative generic format	{baseURI}/{entities}/{ID}/{resource}
Example	{baseURI}/vehicles/123/positions Get all positions for vehicle with id 123.
	{baseURI}/vehicles/456/odometers Get the odometer value for vehicle with id 456.
	{baseURI}/vehicles/456/tirePressures Get the tire pressures of all wheels on the vehicle with id 456.

REQ_04_02_13	If filtering of the response is needed, query parameters shall be used.

NOTE 6 Several query parameters can be added to a request.

NOTE 7 See Table 6.

Table 6 — Examples of filtering responses expresses REQ_04_02_13 and REQ_04_02_01

Normative generic format	{baseURI}/{entities}?{filter}={filterValue}
	{baseURI}/{entities}?{filter}={filterValue}&{filter2}={filterValue2}
Example	{baseURI}/vehicles?ignitionState=on Get all vehicles with ignition on.
	{baseURI}/vehicles/123/positions?startDate=...&endDate=... Get the positions for vehicle with id 123 registered in given date span.

REQ_04_02_14	If sorting is needed, query parameters shall be used.

NOTE 8 See Table 7.

Table 7 — Examples of query parameters expresses REQ_04_02_14 and REQ_04_02_01

Normative generic format	{baseURI}/{entities}?{sorting}={sortingValue}
	{baseURI}/{entities}?{sorting}={sortingValue}&{sorting2}={sortingValue2}
Example	{baseURI}/vehicles?sortField=id&sortOrder=asc

REQ_04_02_15	If selection of subsets of Resources is needed, query parameters shall be used.

NOTE 9 See Table 8.

Table 8 — Examples of query parameters for subsets expresses REQ_04_02_15 and REQ_04_02_01

Normative generic format	{baseURI}/{entities}?{id}={ID}
	{baseURI}/{entities}?{id}={ID}&{id2}={ID2}
Example	{baseURI}/vehicles?id=123&id=124
	{baseURI}/vehicles?id=YS2RX20001754836

Identifiers may come in multiple formats including, but not limited to, VIN or pseudonymized IDs.

REQ_04_02_16	Pseudonymized IDs may be simple numerical IDs, GUIDs or any other alphanumerical scheme.

NOTE 10 See Table 9.

Table 9 — Examples of pseudonymized IDs represented by numerical IDs expresses REQ_04_02_16 and REQ_04_02_01

Normative generic format	{baseURI}/{entities}?{id}={ID}
	{baseURI}/{entities}?{id}={ID}&{id2}={ID2}
Example	{baseURI}/vehicles?id=ce5d5e3d-28bc-475f-8ef7-b5cb9c8039d4&id=f95ce756-42fc-48b2-8873-86553f6df5cc
	{baseURI}/vehicles?id=456

REQ_04_02_17	For large Resource responses pagination may be used.

NOTE 11 See Table 10.

Table 10 — Examples of pagination expresses REQ_04_02_17 and REQ_04_02_01

Normative generic format	{baseURI}/{entities}?start={value}&limit={count}
Example	{baseURI}/vehicles?start=20&limit=10

REQ_04_02_18	A GET request on the returned location may return the total amount of results. If used, it shall be part of the message body using the keyword “exveTotal”.

The following example expresses REQ_04_02_18:

```
{
"results": {
 "exveTotal": "150", ...
}
```

REQ_04_02_19	If wildcards are used, a wildcard (*) shall access all sub-entities or Resources of all parent entities.

NOTE 12 See Table 11.

Table 11 — Examples of wildcards in URIs expresses REQ_04_02_19 and REQ_04_02_01

Normative generic format	{baseURI}/{entities}/*/{entities2}
Example	{baseURI}/vehicles/*/positions Get all positions for all vehicles.
	{baseURI}/vehicles/*/ecus Get all ECU's of all vehicles.

REQ_04_02_20	If wildcards are used, a wildcard (*) may be combined with other filtering, including IDs, to access a smaller number of entities.

NOTE 13 See Table 12.

Table 12 — Examples of wildcards in URIs while filtering by IDs expresses REQ_04_02_20 and REQ_04_02_01

Normative generic format	{baseURI}/{entities}/*/{resource}?{id}={ID}&{id2}={ID2}&{filter}={filterValue}&{-filter2}={filterValue2}
Example	{baseURI}/vehicles/*/odometers?id=456&id=789& startDate=...&endDate=... Get all odometer values from vehicles with id 456 and 789 registered within the given time span.

REQ_04_02_21	For fully anonymized Access, no entity IDs shall be used in the URIs.

NOTE 14 See Table 13.

Table 13 — Examples of an anonymized Access expresses REQ_04_02_21 and REQ_04_02_01

Normative generic format	{baseURI}/{entities}/{entities2}
Example	{baseURI}/vehicles/hazardWarnings?isActive=true

4.3 HTTP Header Fields

REQ_04_03_01	A client shall send a Host header field in all HTTP/1.1 request messages, see RFC 7230[8].

NOTE 1 The server name is the same as the "Host" name as outlined in the extended vehicle URI format; see Table 1.

REQ_04_03_02	The HTTP request header fields "Authorization" shall be present in every client HTTP request.

REQ_04_03_03	The HTTP request header field "Accept" shall identify the media type the client accepts in the response from the server.

REQ_04_03_04	The HTTP response header field "Content-Type" shall be present in every server HTTP response with content.

REQ_04_03_05	The HTTP request header field “Authorization” shall use the “Bearer” authentication scheme to transmit a token.

NOTE 2 The generation, format and use of the token is further specified in ISO 20078-3.

4.4 Media Types

REQ_04_04_01	The Media type: application/json; utf-8 should be supported in HTTP request and response messages

REQ_04_04_02	Media types that may be supported in HTTP request and response messages are: — text/plain; utf-8, — text/xml; utf-8.

REQ_04_04_03	At least one media type shall be selected when requesting data.

4.5 Resource Versioning

REQ_04_05_01	The Resource versioning shall be done on Resource level.

REQ_04_05_02	The API versioning may be done in the URL.

REQ_04_05_03	If Resource versioning is used, Resource versions shall be identified by custom media types included in the request and response header fields “Accept” and “Content-Type”.

EXAMPLE Media types indicating Resource versions, when versioning on Resource level:

— application/x.exve.usecase-resource.positions.v2+json; charset=utf-8,

— application/x.exve.usecase-resource.v1+json ; charset=utf-8.

REQ_04_05_04	The syntax of the custom media type shall be: application/x.exve.{usecase-resource}.{resource version}+ {requested return format}; charset=utf-8

NOTE 1 The use case resource with path is a unique path + name for the Resource type.

NOTE 2 The Resource version is the version of the Resource that the client wants to be returned. It does not have to be the latest available on the server, but the latest the client can handle.

NOTE 3 The requested return format is the format the client wants to have the response in (e.g. json, xml), following REQ_04_05_01.

NOTE 4 The character encoding is utf-8.

REQ_04_05_05	Available versions for each Resource shall be documented in the use case specific Resources.

REQ_04_05_06	If no version is defined in the accept header, the latest version shall be returned.

4.6 Resources and web services

Resources are exposed through web services. In most cases there is a many-to-many relationship between Resources and web services. This subclause outlines how web services and Resources relate to each other and how granting of Resources is reflected when exposing them through web services.

4.6.1 General

REQ_04_06_01	A web service shall include 1 or more Resources.

REQ_04_06_02	A resource may be included in many web services.

REQ_04_06_03	Granting access to a Resource, either directly or through a Container, shall give Access to that Resource only in the web services exposing it.

NOTE 1 If all Resources in a web service are granted, full Access to the web service is given.

NOTE 2 If no Resource in a web service is granted, no Access to the web service is given.

NOTE 3 If a subset of the Resources in a web service is granted, partial Access to the web service is given; there might be cases where no Access can be given, due to how the web service is structured.

REQ_04_06_04	The web service specification shall not be changed in any way, due to granting, denying, ignoring, or revoking Access of included Resources.

NOTE 4 The web service specification handles that only a subset of the included Resources might be granted.

4.6.2 Examples

Table 14 shows examples of how Resources are exposed in web services and how granting of these Resources affects what is accessible.

Table 14 — Resources exposed in web services

Resource: Web Service	Example	Resource	Web service	Grant	Access
1:1	1	A	A	A	A
	2	B	B	—	—
N:1	3	A	AB	A	AB
		B		B	
	4	C	CD	C	C (if possible)
		D		—	
	5	E	EF	—	—
		F		—	
1:N	6	G	G’	G	G’
			G’’		G’’
	7	H	H’	—	—
			H’’		—

Table 14 can be selectively explained by

— Example 2: Resource B is included in web service B. If Resource B is not granted, no Access to web service B is allowed,

— Example 4: Resource C and D are included in the web service CD. If Resource C is granted, but not resource D, access to C using the web service CD is allowed. (Web service CD needs to be structured in a way making this possible),

— Example 6: Resource G is included in both web service G' and G''. If resource G is granted, full access to both web service G' and G'' is allowed.

4.7 Rate Limits

To avoid that clients put excessive load on the server side interface, rate limitation could be imposed on server requests. When rate limitations are applied, a number of HTTP headers are used in order to allow a client to be informed about the current rate limitations for the service.

Rate limits are applied to intervals, in each interval there is a limited amount of requests.

REQ_04_07_01	The Offering Party may apply rate limits.

REQ_04_07_02	If rate limits are applied, then the HTTP header 'X-Rate-Limit-Limit' should be used to indicate the request limit within a given time frame.

REQ_04_07_03	If rate limits are applied, then the HTTP header 'X-Rate-Limit-Remaining' should be used to indicate remaining available number of requests within the given time frame.

REQ_04_07_04	If rate limits are applied, then the HTTP header 'X-Rate-Limit-Reset' should indicate the time in UTC, expressed as seconds since 01.01.1970 (unix timestamp, epoch), when the 'X-Rate-Limit-Remaining' is reset to 'X-Rate-Limit-Limit'.

The following holds as an example, the responses are sent back at 2016-04-06T20:00:00 (1459972800).

NOTE See Table 15.

Table 15 — Examples for different X-Rate-Limit parameters

Resource	X-Rate-Limit-Remaining	X-Rate-Limit-Reset	X-Rate-Limit-Limit
{base_URI}/{resource1}	55	1459973400	60
{base_URI}/{resource2}	2	1459984500	4
{base_URI}/{resource3}	28	1459973460	30

REQ_04_07_05	If rate limits are applied, then the time frames applied for 'X-Rate-Limit-Reset' and the limits for 'X-Rate-Limit-Limit' should be defined by the Offering Party.

REQ_04_07_06	When the rate limit is exceeded, HTTP 429 shall be returned "Too Many Requests".

4.8 HTTP Methods

REQ_04_08_01	The basic HTTP methods POST, GET, PUT and DELETE shall be supported to create, read, update, and delete the Resources where applicable.

NOTE 1 The HTTP method PATCH can be used to enable and ease implementation of use cases.

REQ_04_08_02	The HTTP GET method shall be used to read information from an addressed Resource on the server.

EXAMPLE 1 GET {base URI}/{resources} HTTP/1.1.

REQ_04_08_03	The HTTP POST method shall be used to create a new Resource on the server.

EXAMPLE 2 POST {base URI}/{resources} HTTP/1.1.

REQ_04_08_04	The HTTP PUT method shall be used to change the information for an existing Resource on the server.

NOTE 2 Any examples are provided by a selected use case, due to being highly dependent.

EXAMPLE 3 PUT {base URI}/{resources}/{resourceId} HTTP/1.1.

REQ_04_08_05	The HTTP DELETE method shall be used to delete a resource on the server.

EXAMPLE 4 DELETE {base URI}/{resources}/{resourceId} HTTP/1.1.

REQ_04_08_06	The authorization for the HTTP POST, GET, PUT, and DELETE methods on Resources is managed by the Offering Party.

If use cases run into problems regarding concurrent Access to Resources, *Conditional Requests*[4] can be considered as a solution.

EXAMPLE 5 The ETag and If-Match Headers can be used to detect conflicts.

For partial updates of Resources, the HTTP PATCH method[5] can enable and ease some use cases.

EXAMPLE 6 The two PATCH methods *JSON Merge Patch*[6] and *JavaScript Object Notation (JSON) Patch*[7] can be options to implement PATCH for applicable use cases.

4.9 HTTP Response Status Codes

REQ_04_09_01	Every response message from the web service of the Offering Party shall include a status code indicating the result of the request operation. See RFC 7230[8] chapter 3.

REQ_04_09_02	The status codes outlined in Tables 15, 16, 17 and 18 shall be supported as a minimum set by the web service of the Offering Party.

REQ_04_09_03	Returning a 4xx or 5xx status code on a request shall imply that no further processing of the request will be done at the Offering Party.

Table 16 — Minimum server side supported response status codes on Success

Code	Reason	Description	Justifications
200	Ok	The request succeeded	An ExVe server will always support an OK response.
201	Created	The request completed, and a new resource was created.	An ExVe server will be able to return this code in the instance that a resource was created successfully.
202	Accepted	The request has been accepted for processing, but not completed.	An ExVe server will be able to return this code to indicate that it has begun a processing task, e.g. a data read request.

Table 17 — Minimum server side supported response status codes on Client Error

Code	Reason	Description	Justifications
400	Bad Request	The request could not complete due to malformed syntax.	An ExVe server will use this response to indicate that the request was in some way malformed and should be corrected before a repeat request is made.
401	Unauthorized	This request requires user authentication to proceed.	An ExVe server will use this response if the request of the client is not correctly authenticated.
403	Forbidden	The request was understood, but was refused.	An ExVe server will use this response if the requesting party was not authorized to continue for whatever reason.
404	Not Found	The requested resource was not found.	An ExVe server will use this response to indicate that the requested resource was not available.
406	Not Acceptable	The resource identified by the request is not capable of generating a response that matches the given Accept headers.	An ExVe server will use this response to indicate that a given request was not supported at the level required
429	Too Many Requests	Indicates that the client has sent too many requests to a Resource in a given amount of time ("rate limiting").	An ExVe server will use this response to indicate that too many requests was sent to a specific resource for a given amount of time.

Table 18 — Minimum server side supported response status codes on Server Error

Code	Reason	Description	Justifications
500	Internal Server Error	The server was unable to complete the request due to an unexpected condition or error.	An ExVe server will indicate that the server encountered an error, and the request was not processed.
501	Not Implemented	The server does not support the functionality required	An ExVe server will return this response if a function was requested that did not have implemented functionality.
503	Service Unavailable	The server is unable to service the request due to a temporary unavailability condition.	An ExVe server will return this response if the service was unable to process the request for resource-related reasons.
505	Version not supported	The server does not, or refuses to support the protocol associated with the request.	An ExVe server will support this response, if versioning is supplied.

4.10 Error and Information Messaging

Synchronous and asynchronous REST interactions can fail due to various reasons. To enable the Accessing Party to narrow down the cause, the Offering Party provides suitable error messages. An error consists of an id (exveErrorId) and a short human readable statement (exveErrorMsg).

To avoid confusion, it should be recognized that depending on the interaction pattern, error messaging can be included in unsuccessful requests (e.g., HTTP 4xx) but can also be involved in some HTTP 200 transmissions.

EXAMPLE 1 If in an asynchronous interaction the request fails after some time.

For successful and unsuccessful interactions, the Offering Party can provide additional information with a 'note' statement (exveNote).

REQ_04_10_01	For error messaging the Offering Party shall include a unique identifier in the message body using the keyword “exveErrorId”.

EXAMPLE 2 “exveErrorID”: "7".

REQ_04_10_02	For error messaging the Offering Party shall include a short human-readable statement in the message body using the keyword “exveErrorMsg”.

EXAMPLE 3 “exveErrorMsg”: "Your request timed out (limit: 120s)".

REQ_04_10_03	To unify and simplify error checking, “exveErrorId” and “exveErrorMsg” keywords shall not be present at all, if a request is fully successful.

NOTE 1 exveErrorId and exveErrorMsg are not set to null or left empty. If an Accessing Party finds exveErrorId or exveErrorMsg in a response message body an error occurred.

A partially successful request could return HTTP 200 and error information indicating the status of the request.

REQ_04_10_04	For error messaging the Offering Party should include an error reference in the message body using the keyword “exveErrorRef”.

NOTE 2 The keyword exveErrorRef is represented by a UUID; see ISO 20078-1.

EXAMPLE 4 “exveErrorRef”: "848d8c29-c2e6-4a88-8069-8e4e37454814".

REQ_04_10_05	The Offering Party should include further information in the message body of any response (successful or unsuccessful) by using the keyword “exveNote”.

EXAMPLE 5 “exveNote”: "Please retry upon a random time period".

REQ_04_10_06	The elements: “exveErrorMsg”, and “exveNote” shall be plain string literals.

NOTE 3 These elements are not containing any further (child) elements and are not of type; e.g. array or object.

REQ_04_10_07	The Offering Party may return multiple error messages in the same response.

NOTE 4 When sending multiple error messages an array of error message objects may be used.

Example of a failed asynchronous response using JSON:

```
{
 "dtcReadout": {
 "id": "abcde-12345-ghjke-67474",
 “asyncStatus": "Fail",
 “exveErrorId": “7”,
 “exveErrorRef”: “{UUID}”,
 “exveErrorMsg": " Your request timed out (limit: 120s)",
 “exveNote": "You did not provide a limit for the amount of results; this was automatically set to 150.",
}
}
```

Example of successful JSON request with additional note information:

```
{


"dtcReadout": {
  "id": "abcde-12345-ghjke-67474",
  “asyncStatus": "Complete",
  “exveNote": "You are accessing dtcReadout without versioning information. The API will be upgraded in two month; please see http://{baseURI}/xyz for further information.",
[... data...]
 }
}
```

4.11 Interaction Pattern

4.11.1 Asynchronous

Some functions require an asynchronous interaction pattern (Figure 1). This is for example required when Resources need to be read from the Connected Vehicle rather than being available off-board.

Reading the Connected Vehicle’s current position and speed or the active diagnostics trouble codes could be examples of this. Depending on the architecture and implementation, such operations add a round trip delay that is hard to determine due to the connectivity of the vehicle, e.g. network latency, network coverage or vehicle in wrong mode.

To cope with unpredictable response times and avoid keeping the HTTP session open for a long time, the server will respond directly to the client request. Either with the result of the operation or with a location where the client can retrieve (request again) the status of the operation and finally retrieving the result.

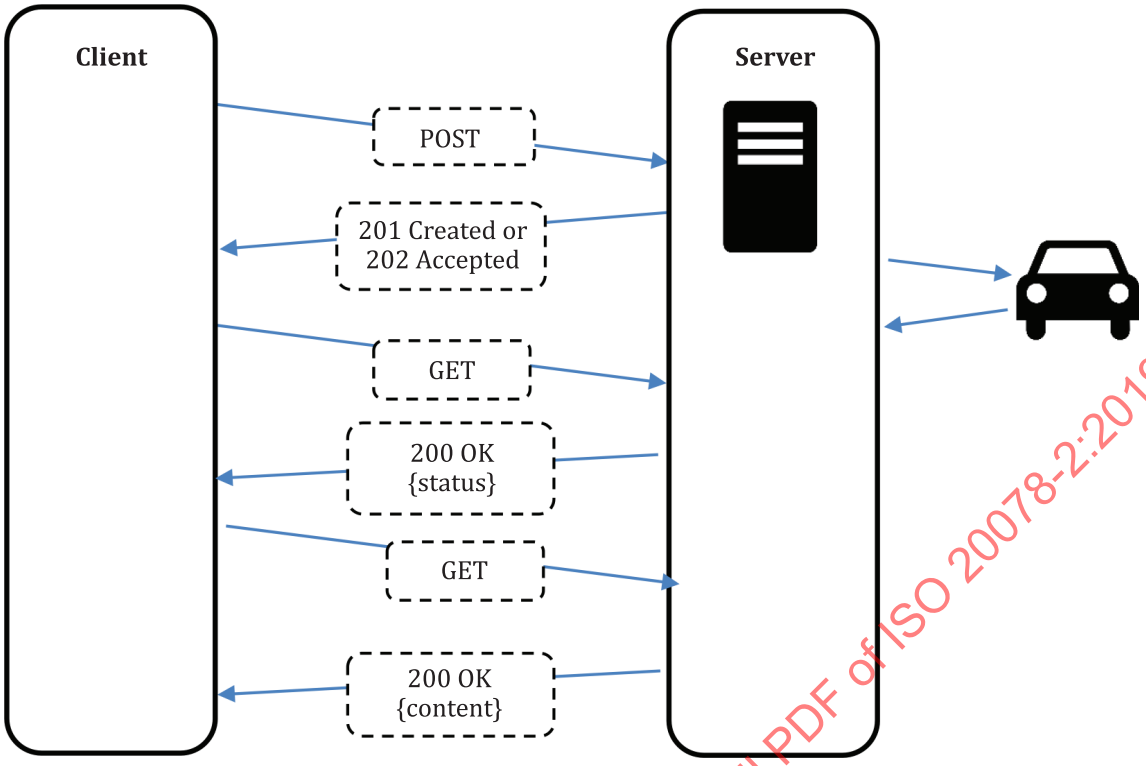

Figure 1 — Asynchronous Interaction pattern

REQ_04_11_01	The asynchronous interaction pattern shall start with an HTTP POST request.

REQ_04_11_02	The POST request should include query parameters and/or message body parameters.

EXAMPLE 1 Request the readout of the active DTCs (Diagnostic Trouble Codes) from a Connected Vehicle: POST {base URI}/vehicles/{VIN}/dtcReadouts?dtcStatus=ACTIVE.

NOTE 1 The query parameters and/or message body parameters are web service specific. Available parameters, their format and expected behaviour are described in the web service documentation.

REQ_04_11_03	If the asynchronous request result is available immediately, the response to a POST request shall be “201 Created” and include the result in the message body.

REQ_04_11_04	If the asynchronous request result is not available immediately, the response to a POST request shall be “202 Accepted” and the status of the request shall be available at an URI provided in the HTTP header “Location”

EXAMPLE 2 A location where the id is typically randomly generated and only valid for this request: {base URI}/vehicles/{VIN}/dtcReadouts/{id}.

NOTE 2 The client polls this URI until the result is available or tries to fetch the result at an estimated time of arrival. The intended/preferred access method (continuous frequent polling or finally try to catch results) and the allowed polling frequency is part of the web service documentation.

REQ_04_11_05	A GET request on the returned location shall return the status of the asynchronous request in the message body using the keyword “asyncStatus”.

REQ_04_11_06	The status of the asynchronous request shall be one of the following values: Pending, InProgress, Complete, Fail

NOTE 3 See Table 19.

Table 19 — List of introduced codes for processing status

Status	Description
Pending	The requested operation has not yet started processing.
InProgress	The requested operation is processing but is not completed.
Complete	The requested operation is complete and has succeeded.
Fail	The requested operation is completed and has failed or timed out.

REQ_04_11_07	A GET request on the returned location should return a recommended waiting time in milliseconds before making the next request. If used, it shall be part of the message body using the keyword “asyncWait”.

REQ_04_11_08	A GET request on the returned location should return the date and time in ISO 8601 format when the asynchronous request is estimated to be completed. If used, it shall be part of the message body using the keyword “asyncEstimatedComplete”.

REQ_04_11_09	A GET request on the returned location should return the estimated progress in percentage of the asynchronous request. If used, it shall be part of the message body using the keyword “asyncProgress”.

REQ_04_11_10	A GET request on the returned location may return the time when the aynchronous request is ended and the related data is made unavailable. If used, it shall be part of the message body using the keyword “asyncRequestEndTime.

REQ_04_11_11	If the asynchronous request is completed, the message body shall contain the result.

REQ_04_11_12	If the asynchronous request has failed, the message body shall contain the reason.

REQ_04_11_13	If the asynchronous request end time has been exceeded, the response shall be a 404 Not found.

Example of a response using JSON when the asynchronous request is in progress:

```
{
"dtcReadout": {
  "id": "abcde-12345-ghjke-67474",
  “asyncStatus": "InProgress",
  “asyncWait”: 10000,
  “asyncEstimatedComple.te”: “2017-05-30T15:35:00Z”,
  “asyncProgress”: 50,
  “asyncRequestEndTime”: “2017-06-30T00:00:00Z”
 }
}
```

Example of a response using JSON when the asynchronous request is completed:

```
{
 "dtcReadout": {
  "id": "abcde-12345-ghjke-67474",
  "asyncStatus": "Complete"
  “asyncRequestEndTime”: “2017-06-30T00:00:00Z”,
  "dtcs": [
  {
   "dtcId": "123456",
   "status": "ACTIVE" ...
```

Example of a response using XML:

```
<dtcReadout>
 <id>abcde-12345-ghjke-67474</id>
 <asyncStatus>InProgress</asyncStatus>
 <asyncWait>10000</asyncWait>
 “asyncRequestEndTime”: “2017-06-30T00:00:00Z”
</dtcReadout>
```

Example of a response using plain text:

```
"id"="abcde-12345-ghjke-67474"
"asyncStatus"="Complete"
“asyncRequestEndTime”: “2017-06-30T00:00:00Z”
```

It is also possible to use this pattern to return intermediate or indicative values, before the final result is available. Therefore, it can be used e.g. in measurement protocols (e.g. show a continuous trend of sensor values on a tester).